

DEPT Dependency Tracker

What Lockdep does and doesn't

2022 . 9 . 13

Byungchul Park || LG Electronics

max.byungchul.park@gmail.com

| Age | Commit message (Expand) | Author | Files | Lines |
|------------|--|----------------|-------|---------|
| 2020-01-24 | rcu: Add basic support for kfree_rcu() batching | Byungchul Park | 4 | -6/+206 |
| 2019-08-01 | rcu: Change return type of rcu_spawn_one_boost_kthread() | Byungchul Park | 1 | -9/+11 |
| 2018-08-30 | rcu: Refactor rcu_{nmi,irq}_{enter,exit}() | Byungchul Park | 1 | -22/+44 |
| 2018-07-12 | rcu: Change return type of rcu_spawn_one_boost_kthread() | Byungchul Park | 1 | -13/+32 |
| 2018-07-12 | rcu: Improve rcu_spawn_one_boost_kthread() text_switch() reporting | Byungchul Park | 3 | -7/+5 |
| 2018-05-15 | rcu: Remove deprecated RCU debugfs tracing code | Byungchul Park | 2 | -12/+5 |
| 2018-05-15 | rcu: Call wake_nocb_leader_defer() with 'FORCE' when nocb_q_count is high | Byungchul Park | 1 | -1/+1 |
| 2018-05-15 | rcu: Inline rcu_preempt_do_callback() into its sole caller | Byungchul Park | 2 | -11/+1 |
| 2018-02-20 | srcu: Remove dead code in srcu_gp_end() | Byungchul Park | 1 | -2/+0 |
| 2017-11-14 | vhost: Use srcu generation in vhost_scsi to prevent race | Byungchul Park | 1 | -2/+2 |
| 2017-10-31 | irq/vhost: Use srcu generation in vhost_scsi to prevent race | Byungchul Park | 1 | -5/+1 |
| 2017-10-26 | block, locking/lockdep: Associate lockdep with rcu_gp_end() | Byungchul Park | 3 | -11/+23 |
| 2017-10-25 | workqueue: Remove RCU dependency from lockdep initialization | Byungchul Park | 2 | -18/+5 |
| 2017-10-25 | sched/completions: Add support for initializing completions with lockdep_map | Byungchul Park | 1 | -0/+14 |
| 2017-10-25 | locking/lockdep: Introduce CONFIG_BOOTPARAM_LOCKDEP_CROSSRELEASE_FULLSTACK=y | Byungchul Park | 2 | -0/+19 |
| 2017-10-25 | locking/lockdep: Remove the BROKEN flag from CONFIG_LOCKDEP_CROSSRELEASE and ... | Byungchul Park | 1 | -2/+2 |
| 2017-10-25 | locking/lockdep: Add a boot parameter allowing unwind in cross-release and di... | Byungchul Park | 2 | -2/+20 |
| 2017-10-25 | locking/lockdep, sched/completions: Change the prefix of lock name for comple... | Byungchul Park | 1 | -2/+2 |
| 2017-10-25 | locking/lockdep: Provide empty lockdep_map structure for !CONFIG_LOCKDEP | Byungchul Park | 1 | -0/+5 |
| 2017-09-06 | mm/vmalloc.c: don't reinvent the wheel but use existing llist API | Byungchul Park | 1 | -6/+4 |
| 2017-09-06 | bcache: Don't reinvent the wheel but use existing llist API | Byungchul Park | 1 | -13/+2 |
| 2017-08-28 | fput: Don't reinvent the wheel but use existing llist API | Byungchul Park | 1 | -7/+5 |
| 2017-08-28 | namespace.c: Don't reinvent the wheel but use existing llist API | Byungchul Park | 1 | -5/+3 |
| 2017-08-17 | locking/lockdep: Rename CONFIG_LOCKDEP_COMPLETE to CONFIG_LOCKDEP_COMPLETIONS | Byungchul Park | 2 | -6/+6 |
| 2017-08-17 | locking/lockdep: Reword title of LOCKDEP_CROSSRELEASE config | Byungchul Park | 1 | -1/+1 |
| 2017-08-17 | locking/lockdep: Make CONFIG_LOCKDEP_CROSSRELEASE part of CONFIG_PROVE_LOCKING | Byungchul Park | 1 | -5/+2 |
| 2017-08-14 | locking/lockdep: Fix the rollback and overwrite detection logic in crossrelease | Byungchul Park | 1 | -6/+6 |
| 2017-08-14 | locking/lockdep: Add a comment about crossrelease_hist_end() in lockdep_sys_e... | Byungchul Park | 1 | -0/+4 |

Interest : Task Scheduler, RCU, Locking

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/log/?qt=author&q=byungchul>

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds A.

spin_lock A

Lock / unlock balance

Lock and unlock must be used in pairs.

// Terminates the task without releasing A.

exit

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

// Holds A.

spin_lock A

...

// Terminates the task without releasing A.

exit

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

```
// Holds A.
```

```
spin_lock A
```

```
...
```

```
// Terminates the task without releasing A.
```

```
exit
```

REPORT!

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

// Holds A.

spin_lock A

...

// Returns to user without releasing A.

return to user

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds A.

spin_lock A

...

// Returns to user without releasing A.

return to user

REPORT!

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

```
// Holds spin lock A.
```

```
spin_lock A
```

Relation between lock types

```
// Holds mutex lock B.
```

```
mutex_lock B  
...  
mutex_unlock B  
spin_unlock A
```

mutex_lock shouldn't be acquired in a spin_lock critical section.

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds spin lock A.

spin_lock A

...

// Holds mutex lock B.

mutex_lock B

...

mutex_unlock B

spin_unlock A

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds spin lock A.

spin_lock A

...

// Holds mutex lock B.

mutex_lock B

...

mutex_unlock B

spin_unlock A

REPORT!

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

```
// Holds raw spin lock A.
```

```
raw_spin_lock A
```

Relation between lock types

```
// Holds spin lock B.
```

**spin_lock shouldn't be acquired in a
raw_spin_lock critical section.**

```
...
```

```
spin_unlock B
```

```
raw_spin_unlock A
```

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds raw spin lock A.

raw_spin_lock A

...

// Holds spin lock B.

spin_lock B

...

spin_unlock B

raw_spin_unlock A

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds raw spin lock A.

raw_spin_lock A

...

// Holds spin lock B

spin_lock B

...

spin_unlock B

raw_spin_unlock A

REPORT!

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

```
// Holds A.
```

```
spin_lock A
```

Nest lock usage

```
// Holds B with nest lock A.
```

Nest lock API should be called with the nest lock held.

```
...
```

```
spin_unlock B
```

```
spin_unlock A
```


Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds A.

spin_lock A

...

// Holds B with nest lock A.

spin_lock_nest B, n=A

...

spin_unlock B

spin_unlock A

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

// Holds B with nest lock A.

spin_lock_nest B, n=A

...

spin_unlock B

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

```
// Holds B with nest lock A
```

```
spin_lock_nest B, n=A
```

```
...
```

```
spin_unlock B
```

REPORT!

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

RCU read forbidden context

Usage from illegal states

RCU read forbidden context e.g. idle
shouldn't have RCU read sections.

rcu_read_unlock

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

RCU read forbidden context

// Specifies a RCU read section.

rcu_read_lock

...

rcu_read_unlock

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

RCU read forbidden context

// Specifies a RCU read section.

rcu_read_lock

...

rcu_read_unlock

REPORT!

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Specifies a RCU read section.

Proper use of the APIs

`rcu_dereference_protected` is supposed to be used with updater lock held.

`rcu_dereference_protected` A `rcu_dereference` is supposed to be used in RCU read section.

`rcu_read_unlock`

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Specifies a RCU read section.

rcu_read_lock

...

// Calls without necessary lock held.

rcu_dereference_protected A

...

rcu_read_unlock

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Specifies a RCU read section.

rcu_read_lock

...

// Calls without necessary lock held.

rcu_deference_protected A

...

rcu_read_unlock

REPORT!

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Specifies a RCU read section.

`rcu_read_lock`

Delimitation for read section

`rcu_read_unlock`

RCU read APIs should be used in a section delimited by `rcu_read_lock` / `rcu_read_unlock`.

`list_for_each_entry_rcu`

...

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Specifies a RCU read section.

rcu_read_lock

...

rcu_read_unlock

...

// Traverses RCU list out of read section.

list_for_each_entry_rcu

...

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Specifies a RCU read section.

rcu_read_lock

...

rcu_read_unlock

...

// Traverses RCU list out of read section.

list_for_each_entry_rcu

...

REPORT!

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

```
// Holds A.
```

```
spin_lock A
```

Lock acquisition order

```
// Holds B while holding A.
```

```
spin_lock B
```

```
spin_unlock B
```

```
spin_unlock A
```

```
// Holds B.
```

```
spin_lock B
```

```
...
```

```
// Holds A while holding B.
```

```
spin_lock A
```

```
spin_unlock A
```

```
spin_unlock B
```

All the lock acquisition order should be kept across the kernel code.

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds A.

spin_lock A

...

// Holds B while holding A.

spin_lock B

...

spin_unlock B

spin_unlock A

// Holds B.

spin_lock B

...

// Holds A while holding B.

spin_lock A

...

spin_unlock A

spin_unlock B

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

// Holds A.

spin_lock A

...

// Holds B while holding A

spin_lock B

...

spin_unlock B

spin_unlock A

// Holds B.

spin_lock B

...

// Holds A while holding B.

spin_lock A

...

spin_unlock A

spin_unlock B

REPORT!

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

```
// Holds A.
```

```
spin_lock A
```

```
...
```

```
// Holds A and is holding A
```

Lock recursive acquisition

```
spin_lock A
```

The same lock shouldn't be acquired twice in a single context.

```
spin_unlock A
```

```
...
```

```
spin_unlock A
```

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

// Holds A.

spin_lock A

...

// Holds A while holding A.

spin_lock A

...

spin_unlock A

...

spin_unlock A

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

```
// Holds A.
```

```
spin_lock A
```

```
...
```

```
// Holds A while holding A.
```

```
spin_lock A
```

```
...
```

```
spin_unlock A
```

```
...
```

```
spin_unlock A
```

REPORT!

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

interrupt context

```
spin_lock_irq A
```

```
...
```

```
irq_safe_usage A
```

Irq safe APIs should be used if the lock is used in either process or interrupt context.

```
// Holds A with irq disabled.
```

```
spin_lock_irq A
```

```
...
```

```
spin_unlock_irq A
```

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

interrupt context

`spin_lock_irq A`

...

`spin_unlock_irq A`

process context

// Holds A with irq disabled.

`spin_lock_irq A`

...

`spin_unlock_irq A`

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

interrupt context

`spin_lock A`

...

`spin_unlock A`

process context

// Holds A with irq enabled.

`spin_lock A`

...

`spin_unlock A`

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

process context

// Holds A with irq enabled.

spin_lock A

...

interrupt context

spin_lock A

...

spin_unlock A

spin_unlock A

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

process context

// Holds A with irq enabled.

spin_lock A

...

interrupt context

spin_lock A

...

spin_unlock A

spin_unlock A

REPORT!

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

Lockdep is a quite good tool
for checking these.

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

```
mutex_lock A
```

```
...
```

```
// Waits for event B.
```

```
wait_for_event B
```

```
...
```

```
// Releases A.
```

```
mutex_unlock A
```

```
// Waits for A to be released.
```

```
mutex_lock A
```

```
...
```

```
mutex_unlock A
```

```
...
```

```
// Wakes up wait_for_event B.
```

```
event B
```

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

```
mutex_lock A
```

```
...
```

```
// Waits for event B.
```

```
wait_for_event B
```

```
...
```

```
// Releases A.
```

```
mutex_unlock A
```

```
// Waits for A to be released.
```

```
mutex_lock A
```

```
...
```

```
mutex_unlock A
```

```
...
```

```
// Wakes up wait_for_event B.
```

```
event B
```

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

```
mutex_lock A
```

```
...
```

```
// Waits for event B.
```

```
wait_for_event B
```

```
...
```

```
// Releases A.
```

```
mutex_unlock A
```

```
// Waits for A to be released.
```

```
mutex_lock A
```

```
...
```

```
mutex_unlock A
```

```
...
```

```
// Wakes up wait_for_event B.
```

```
event B
```

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

`mutex_lock A`

`...`

`...`

`// Releases A.`

`mutex_unlock A`

`// Waits for A to be released.`

`mutex_lock A`

`...`

`mutex_unlock A`

`...`

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

```
mutex_lock A
```

```
...
```

```
// Releases A.
```

```
mutex_unlock A
```

```
// Waits for A to be released.
```

```
mutex_lock A
```

```
...
```

```
mutex_unlock A
```

```
...
```

REPORT???

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

What makes the case **missed**?

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

How does it track dependency?

Define Dependency

Lock A

Lock B (while holding A.)

= A depends on B. ($A \rightarrow B$)

Example

// Holds A.

spin_lock A

...

// Holds B while holding A.

spin_lock B

...

spin_unlock B

spin_unlock A



Build Dependency Graph

Example

// Holds B.

spin_lock B

...

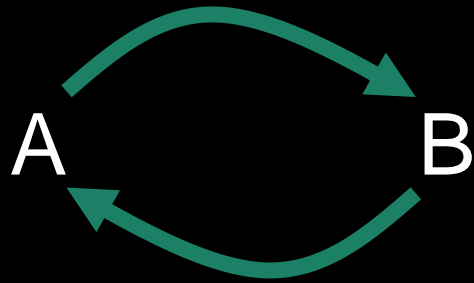
// Holds A while holding B.

spin_lock A

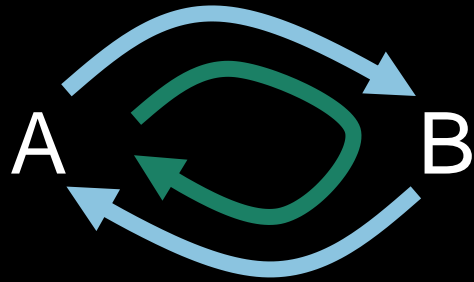
...

spin_unlock A

spin_unlock B

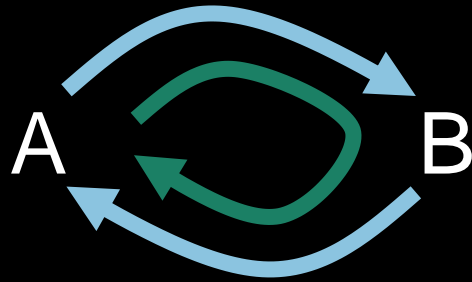


Build Dependency Graph

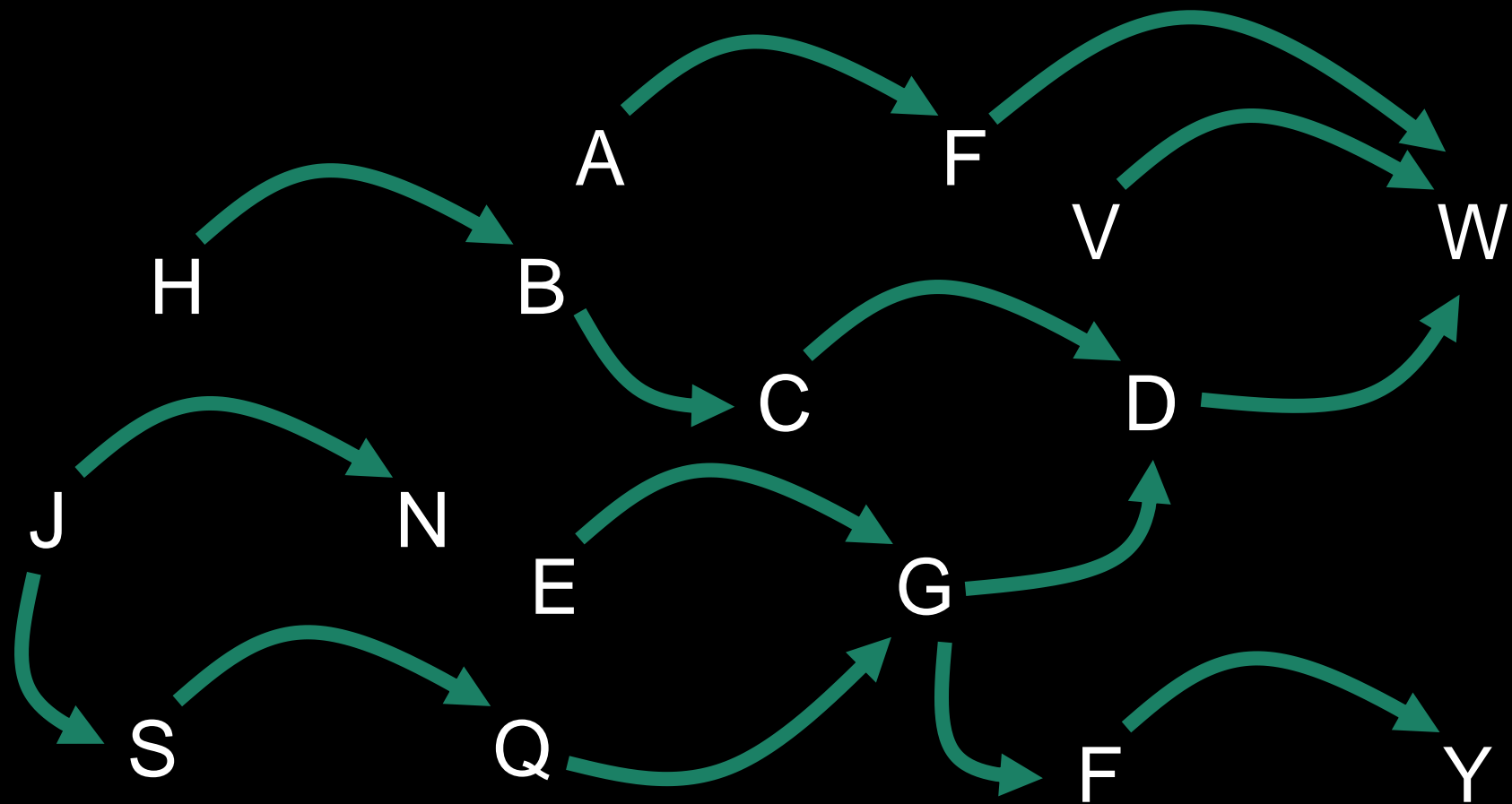


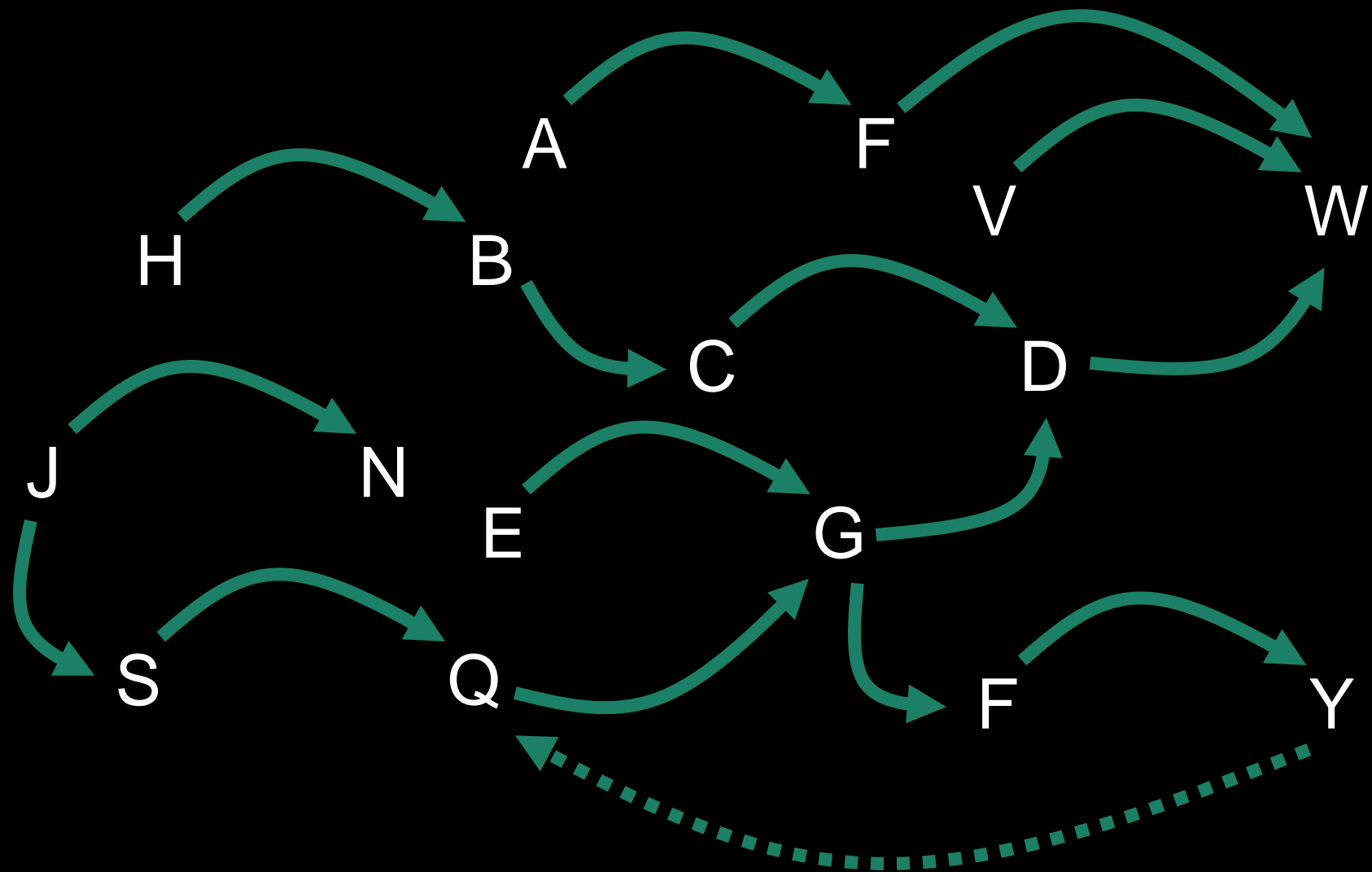
Circular Dependency

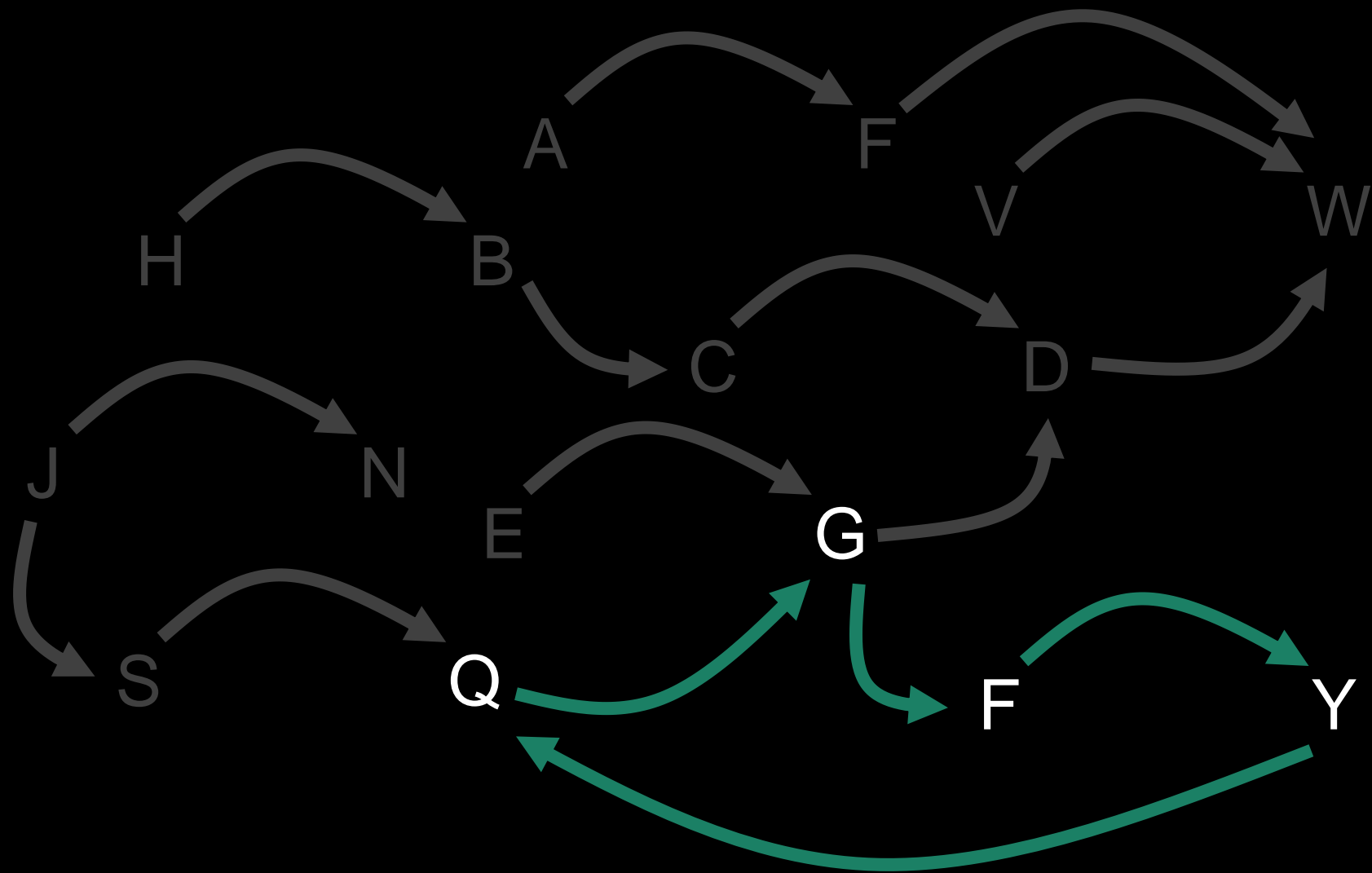
REPORT!

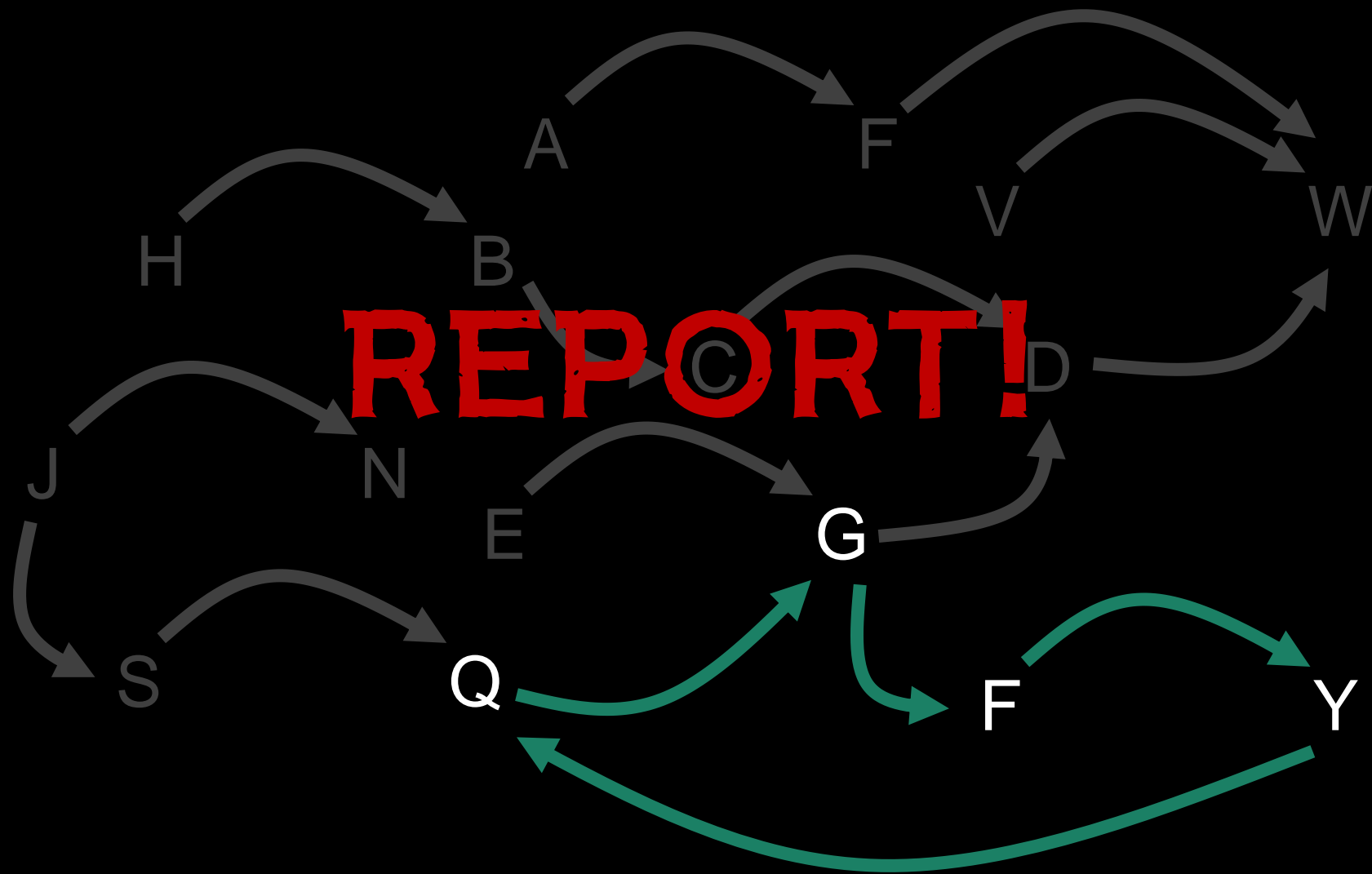


Circular Dependency









Is a deadlock caused by
incorrect lock acquisition order?

Is a deadlock caused by
incorrect lock acquisition order?

Partially yes.

A deadlock is caused by
Waiters that cannot be woken up.

A deadlock is caused by
Events that are not reachable.

Deadlock detection tool should focus on
waits and events themselves.

Define Dependency

Lock A

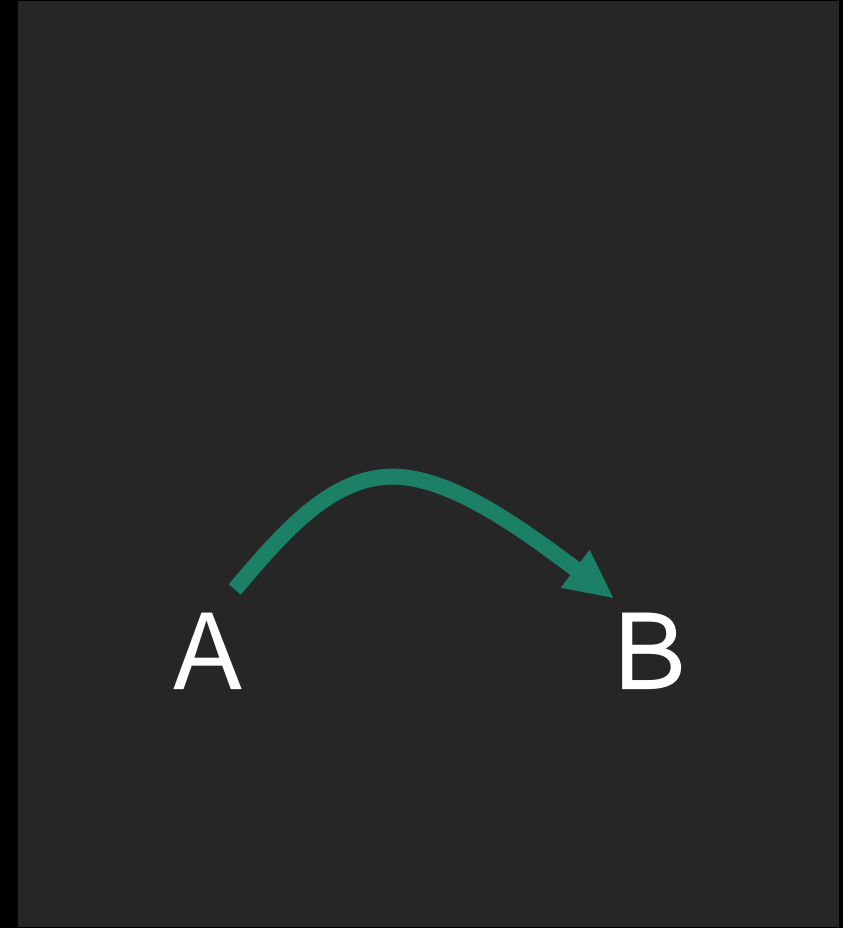
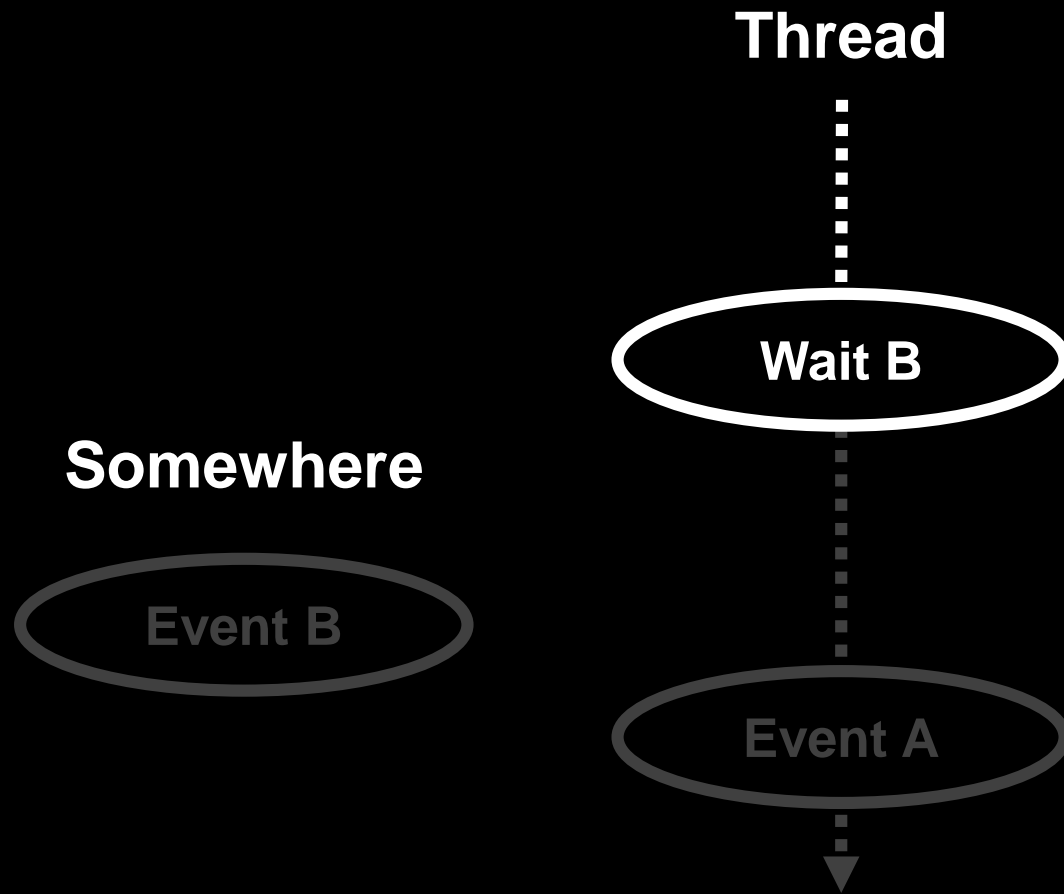
Lock B (while holding A.)

= A depends on B. ($A \rightarrow B$)

Redefine Dependency

Event A occurrence depends on
Event B occurrence.

= A depends on B. ($A \rightarrow B$)



Event A occurrence depends on Event B occurrence.

Example

mutex_lock A

...

// Waits for event B.

wait_for_event B

...

// Depends on event B.

mutex_unlock A



Build Dependency Graph

Example

// Waits for A to be released.

mutex_lock A

...

mutex_unlock A

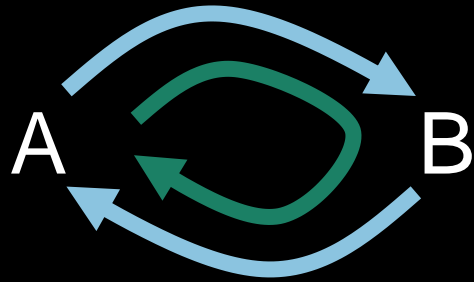
...

// Depends on releasing A.

event B

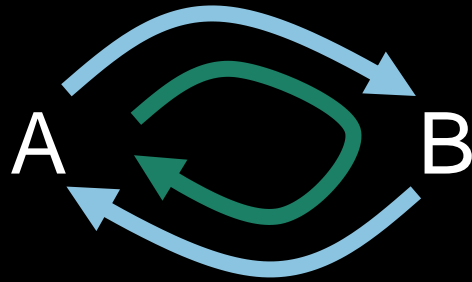


Build Dependency Graph



Circular Dependency

REPORT!



Circular Dependency

REWIND ...

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds A.

spin_lock A

...

// Holds B while holding A.

spin_lock B

...

spin_unlock B

spin_unlock A

// Holds B.

spin_lock B

...

// Holds A while holding B.

spin_lock A

...

spin_unlock A

spin_unlock B

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

// Holds A.

spin_lock A

...

// Holds B while holding A.

spin_lock B

...

spin_unlock B

spin_unlock A

// Holds B.

spin_lock B

...

// Holds A while holding B.

spin_lock A

...

spin_unlock A

spin_unlock B

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

// Holds A.

spin_lock A

...

// Holds A while holding A.

spin_lock A

...

spin_unlock A

...

spin_unlock A

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

// Holds A.

spin_lock A

...

// Holds A while holding A.

spin_lock A
waiter A

...

spin_unlock A

...

spin_unlock A
event A

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

process context

// Holds A with irq enabled.

spin_lock A

...

interrupt context

spin_lock A

...

spin_unlock A

spin_unlock A

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

process context

// Holds A with irq enabled.

spin_lock A

...

interrupt context

waiter A
spin_lock A

...

spin_unlock A

event A
spin_lock A

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

```
mutex_lock A
```

```
...
```

```
// Waits for event B.
```

```
wait_for_event B
```

```
...
```

```
// Releases A.
```

```
mutex_unlock A
```

```
// Waits for A to be released.
```

```
mutex_lock A
```

```
...
```

```
mutex_unlock A
```

```
...
```

```
// Wakes up wait_for_event B.
```

```
event B
```

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

```
mutex_lock A
```

```
...
```

```
// Waits for event B.
```

```
wait_for_event B
```

```
...
```

```
// Releases A.
```

```
mutex_unlock A
```

```
// Waits for A to be released.
```

```
waiter A
```

```
...
```

```
mutex_unlock A
```

```
...
```

```
// Wakes up wait_for_event B.
```

```
event B
```

Lock usage correctness

Lock / unlock balance

Relation between lock types

Nest lock usage

RCU usage correctness

Usage from illegal states

Proper use of the APIs

Delimitation for read section

Lock dependency correctness

Lock acquisition order

Lock recursive acquisitions

Irq safe usage

Lockdep is a great tool
for checking these two.

Lock usage correctness

- Lock / unlock balance

- Relation between lock types

- Nest lock usage

RCU usage correctness

- Usage from illegal states

- Proper use of the APIs

- Delimitation for read section

Lock dependency correctness

- Lock acquisition order

- Lock recursive acquisitions

- Irq safe usage

Lockdep is not the best tool
for this purpose.

Lock usage correctness

- Lock / unlock balance
- Relation between lock types
- Nest lock usage

RCU usage correctness

- Usage from illegal states
- Proper use of the APIs
- Delimitation for read section

Lock dependency correctness

- Lock acquisition order
- Lock recursive acquisitions
- Irq safe usage

So I developed a new tool named **DEPT** to replace the 3rd part to deal with dependency in **the best way**.

Lockdep is not the best tool for this purpose.

Lock usage correctness

- Lock / unlock balance
- Relation between lock types
- Nest lock usage

RCU usage correctness

- Usage from illegal states
- Proper use of the APIs
- Delimitation for read section

Lock dependency correctness

- Lock acquisition order
- Lock recursive acquisitions
- Irq safe usage

So I developed a new tool named
DEPT to replace the 3rd part to deal
with dependency in the best way.

DEPT is the best tool
for this purpose.

DEPT detects situations where
waiters cannot be woken up.

DEPT detects situations where
events are not reachable.

Pros

Works with all types of wait.

Supports multiple reports efficiently.

Provides easy APIs to annotate wait / event.

Cons

Just started so still has false positives.

LKML Discussion

<https://lkml.org/lkml/2022/5/4/218>

Github Repository

<https://github.com/lgebyungchulpark/linux-dept>

WANTED